

Distributed Data Processing with Spring Cloud Data Flow

Kenny Bastani

Spring Developer Advocate



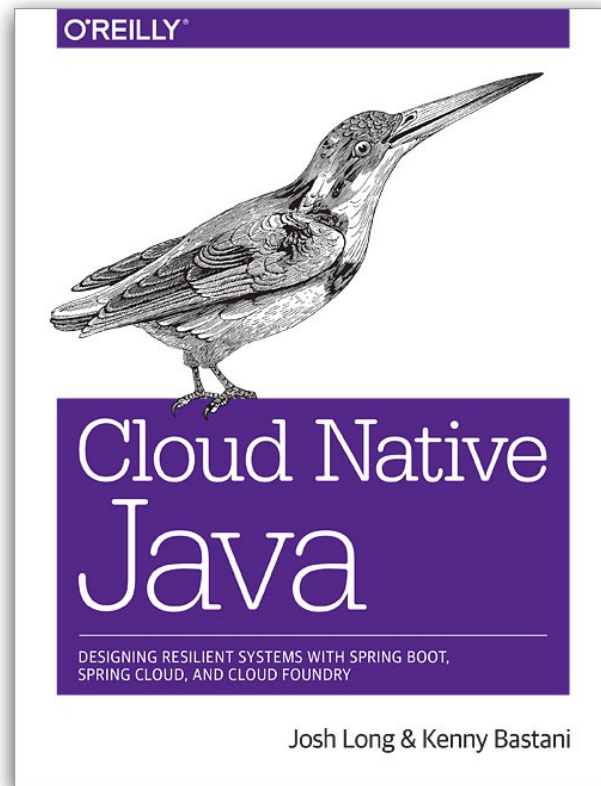
VISIT...

LANZAROTE
Caliente.COM

Speaker Intro

Kenny Bastani

Pivotal®



Agenda

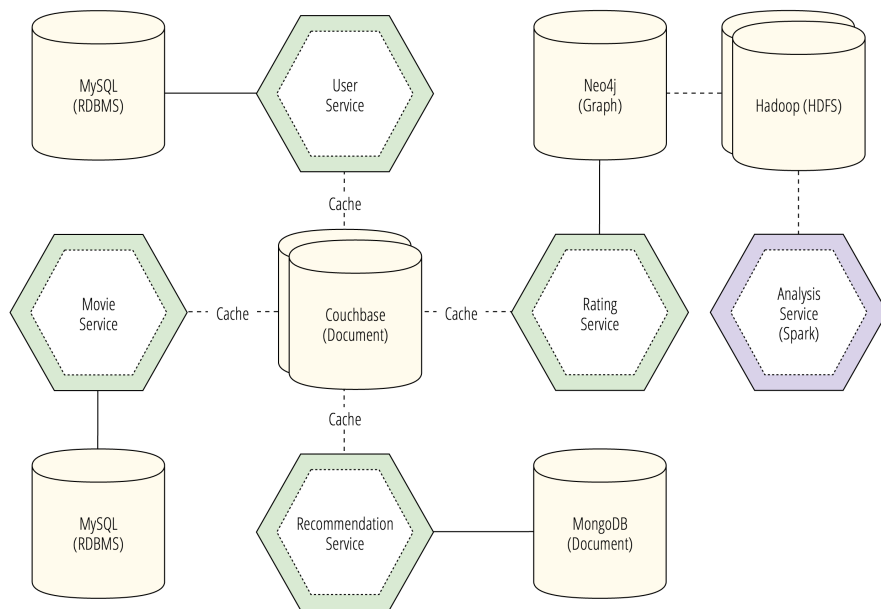
1	Agenda
2	Microservices
3	What is Spring Boot?
4	What is Spring Cloud?
5	Lattice - Cloud Native Platform
6	Spring Cloud Data Flow
7	Streaming Analytics Example (Twitter)
9.	Demo

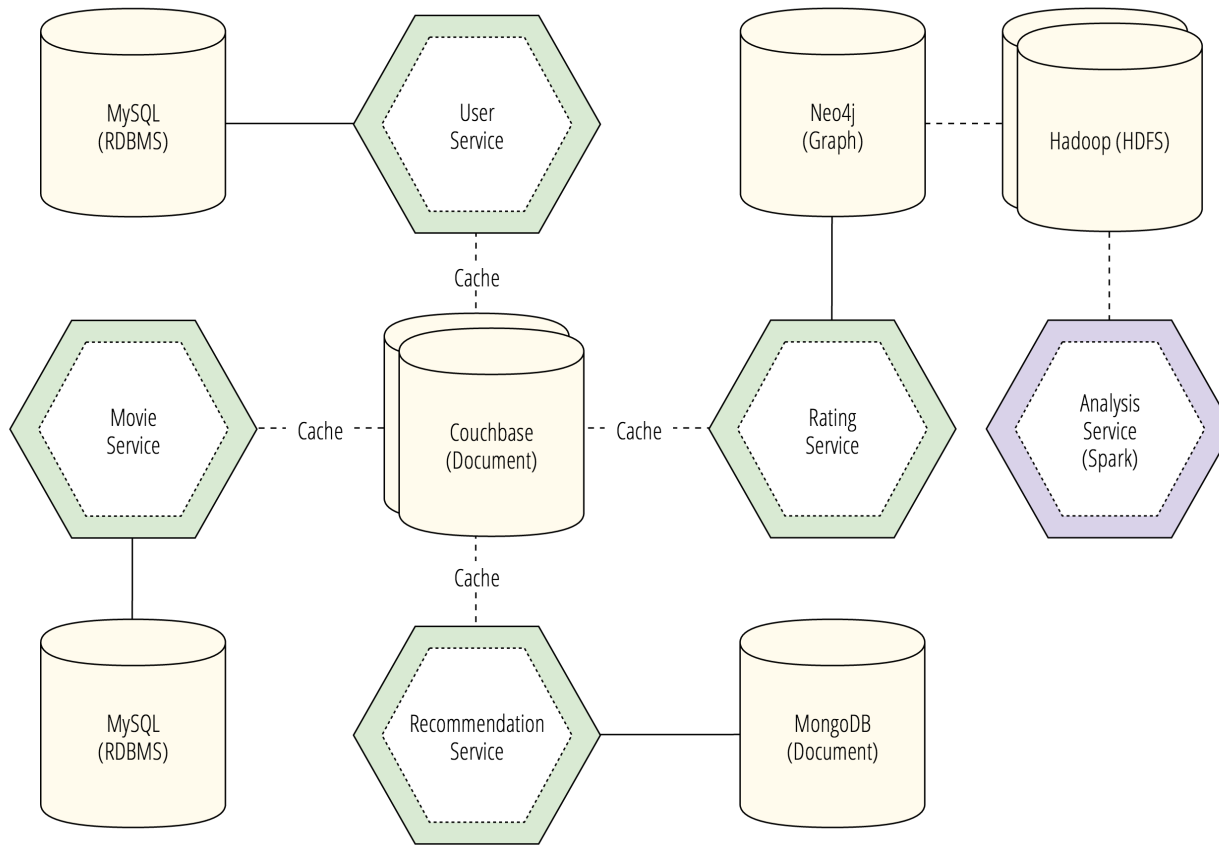
Microservices

“Kind of like a head ache, but more of them.”

Quick Explanation of Microservices

- Each team gets one database and one service
- Shared caches are platform provided services that are shared for consistency

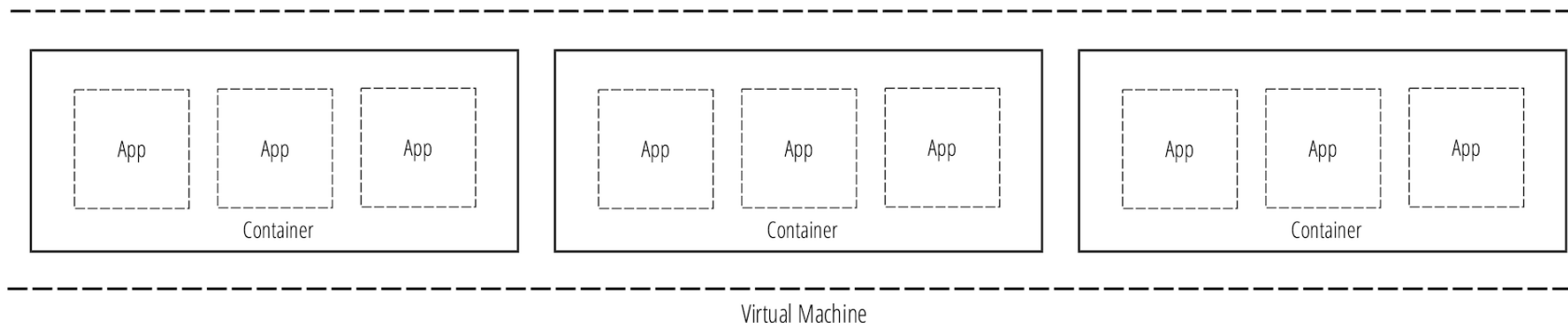




Cloud Native Microservices

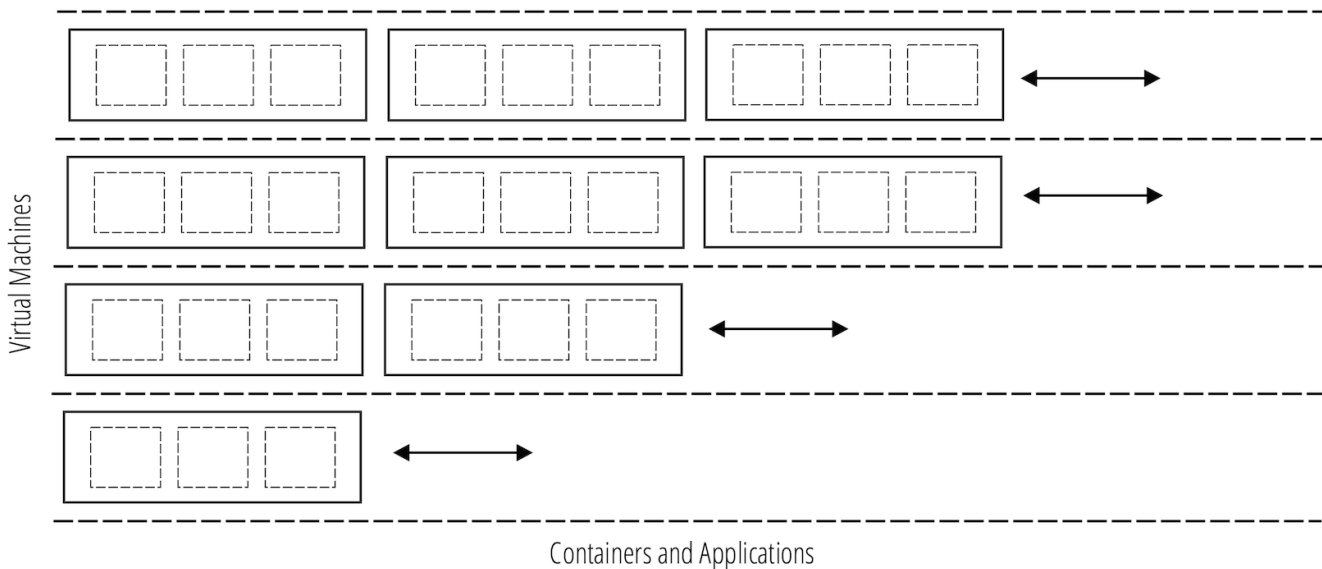
Cloud-native Microservice Deployment

- Each microservice can be containerized with their application dependencies
- Containers get scheduled on virtual machines with an allotted resource policy



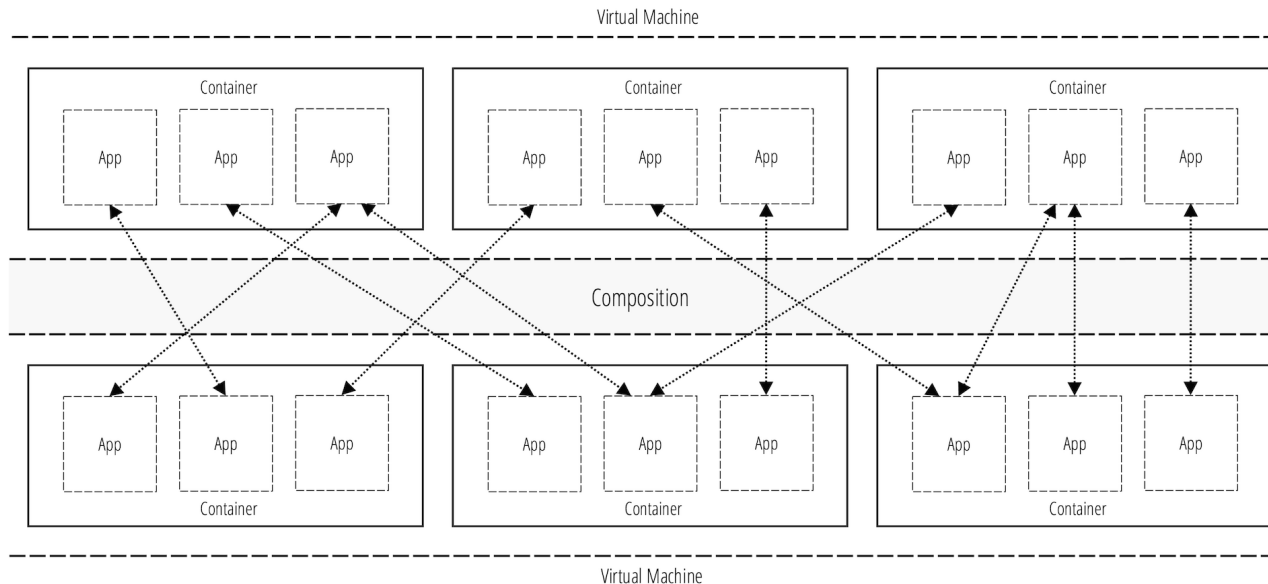
Auto-scaling

- An elastic runtime handles auto-scaling of VMs with cloud providers
- Microservices should be load balanced vertically and not horizontally



Composition & Orchestration

- Each microservice needs to communicate outside containers
- Service discovery provides an automatic method for finding other service dependencies



Spring Boot

A JVM micro-framework for building
microservices

What is Spring Boot?



Phil Webb
@phillip_webb



Following

For those on reddit.com/r/java/ saying it's Spring Boot is "the framework for a framework" here's a diagram:



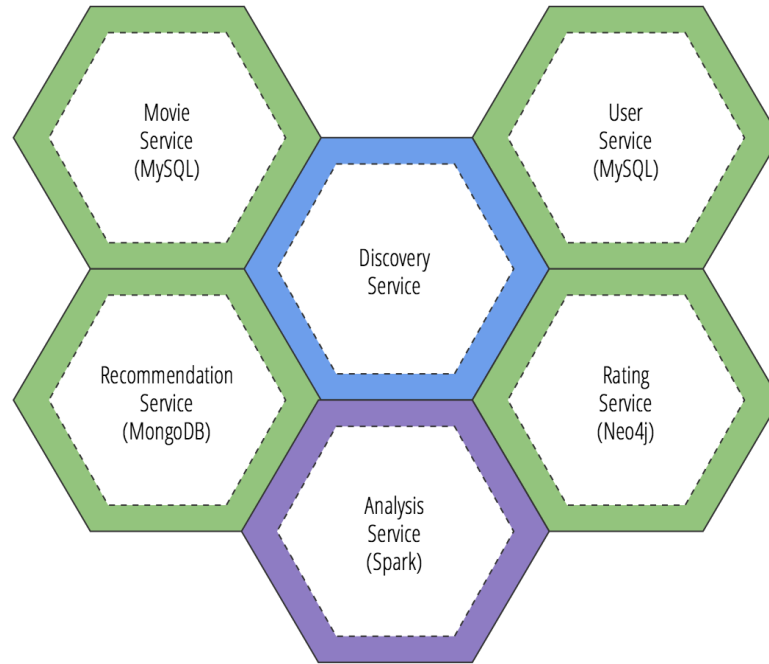
RETWEETS
111

FAVORITES
68



7:54 PM - 8 Sep 2015

Spring Boot Roles



Automatic Configuration

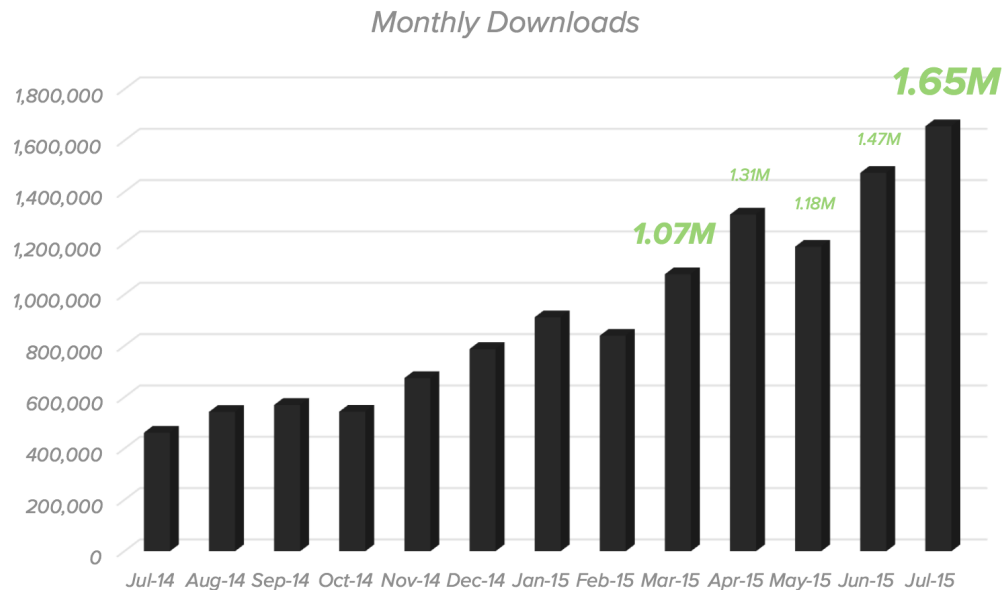
- An application class is annotated with `@SpringBootApplication`
- Additional annotations are added to indicate the role of the Spring Boot application

```
@SpringBootApplication
@EnableSidecar
public class Application {
    @Autowired
    private RepositoryRestMvcConfiguration restConfiguration;

    public static void main(String[] args) {
        new SpringApplicationBuilder(Application.class).web(true).run(args);
    }

    @PostConstruct
    public void postConstructConfiguration() {
        restConfiguration.objectMapper().registerModule(new Jackson2HalModule());
    }
}
```

Spring Boot for Microservices



Spring Cloud

A toolset designed for building distributed systems

What is Spring Cloud?

- Spring Cloud provides a way to turn Spring Boot microservices into distributed applications



Phil Webb

@phillip_webb



Following

For those on reddit.com/r/java/ saying it's Spring Boot is "the framework for a framework" here's a diagram:



RETWEETS
111

FAVORITES
68

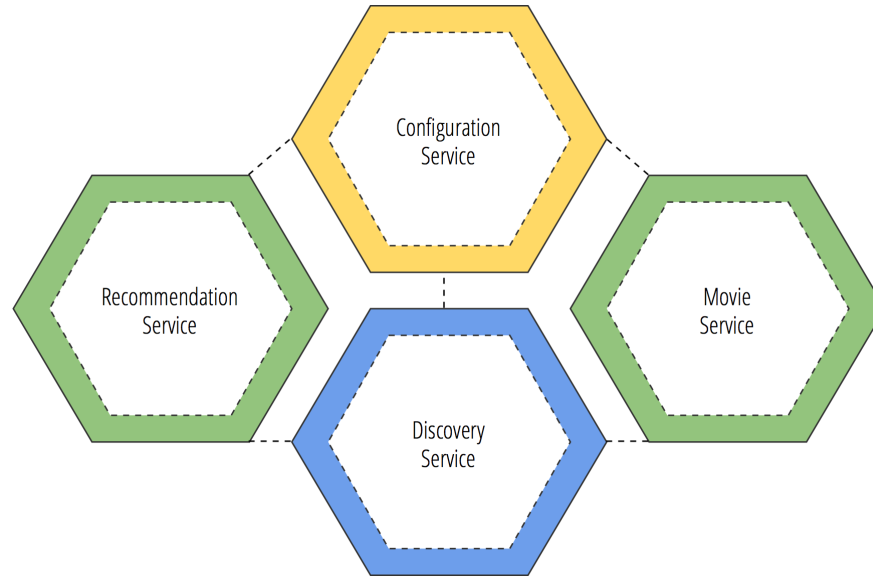


7:54 PM - 8 Sep 2015

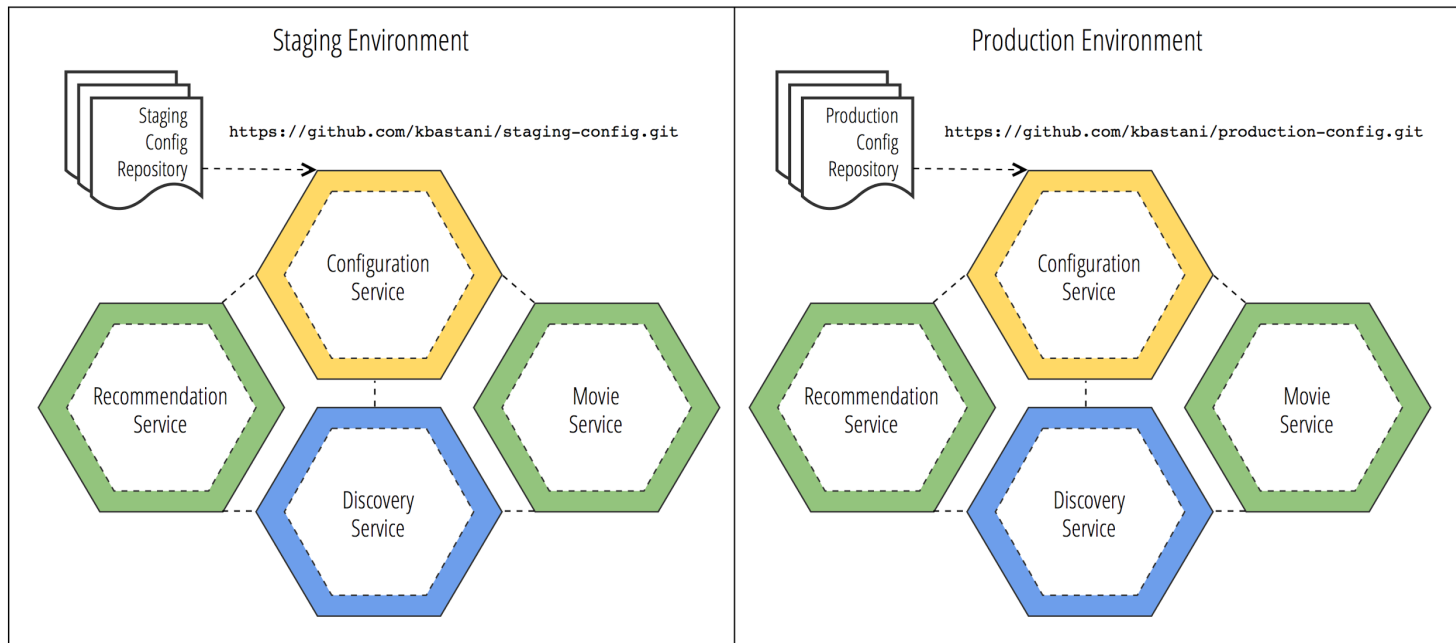
What is Spring Cloud?

- * Service Discovery
- * API Gateway
- * Circuit Breakers
- * Distributed Tracing

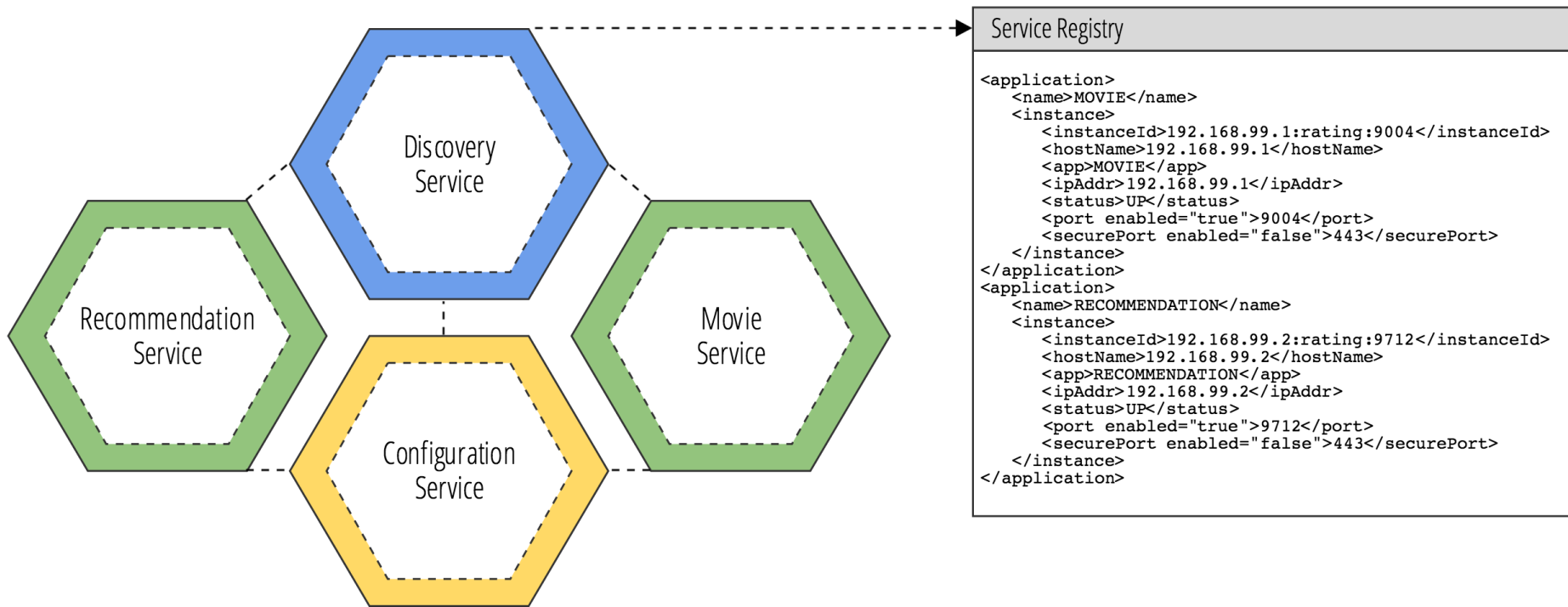
Service Discovery & Configuration Service



Configuration Service



Service Discovery



API Gateway



But what about
deployments?



Lattice

A cloud-native platform for deploying and scaling containers in production

Containers, containers, containers

- Lattice helps you manage Docker container deployments on clusters of VMs
- Choose the cloud provider you want, deploys containers from Docker hub



Spring Cloud Data Flow

Java microservices that process data in a pipeline

Spring Cloud Data Flow

- What is Spring Cloud Data Flow?
 - Spring Cloud Data Flow is a data processing pipeline that uses Spring Boot microservices
 - Each Spring Boot microservice takes in a message and produces a message, containing the data that you're processing

How do we design a data processing pipeline?

Start with understanding your source data

- What do you want to measure?
 - *Trending analytics in real time*

Understand the function of your pipeline

- What does an individual message contain?
 - *A single tweet*

Understand what you want to measure

- What do you want to filter?
 - *A set of hash tags in the body of a Tweet*

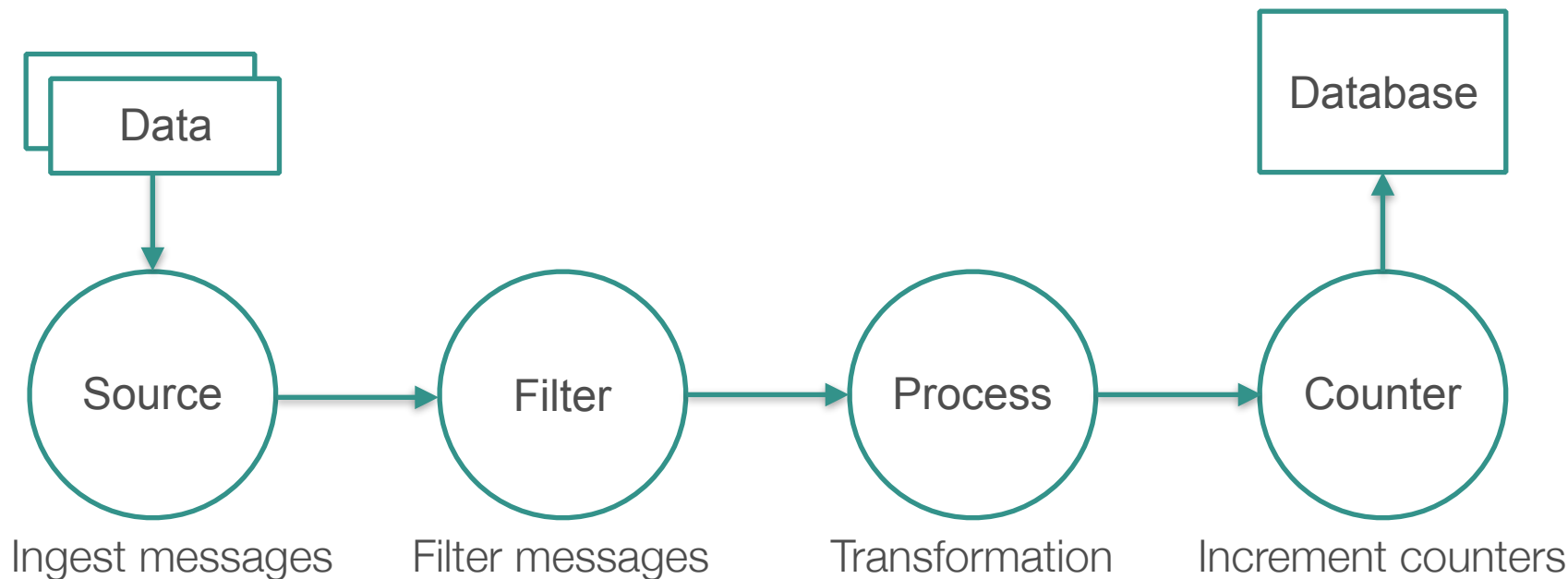
Understand the outputs

- What do I want to measure over time?
 - *The velocity of hash tag counts from tweets every second*

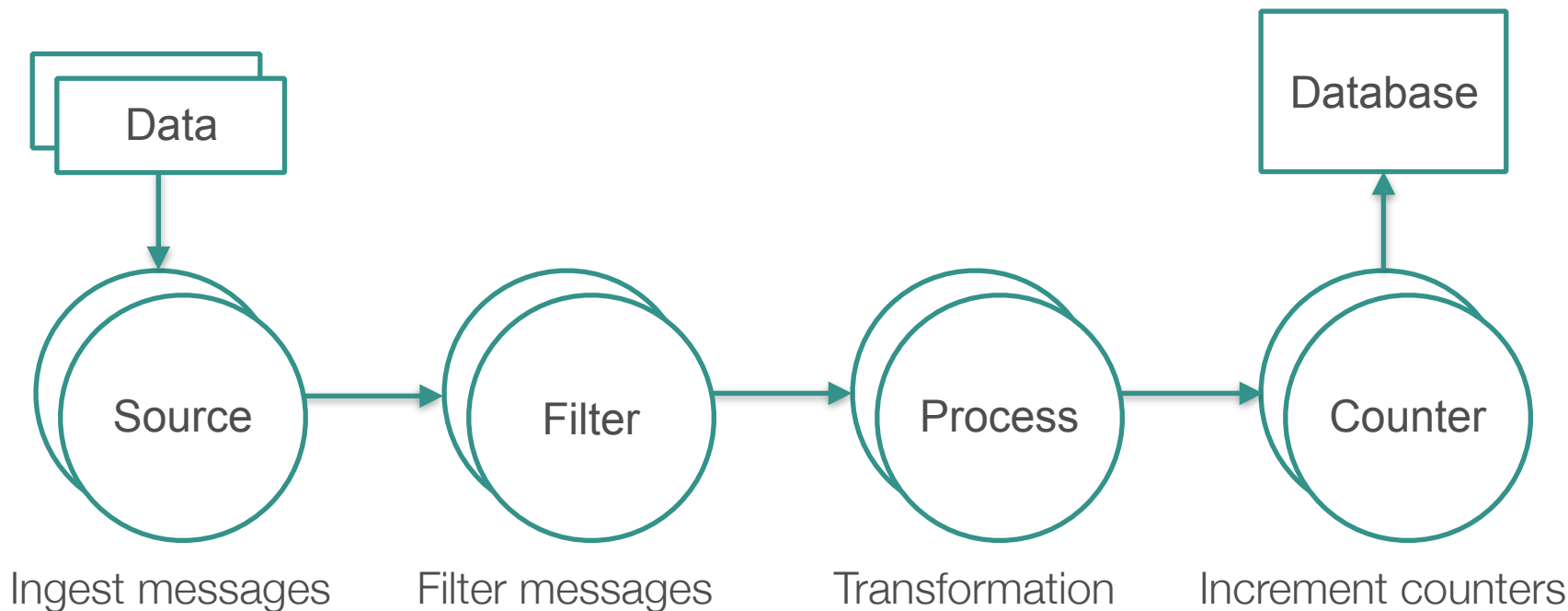
What is the result of our measurements?

- A graph that shows the velocity of each tweet and its hash tags over time
- Real time streaming analytics so we can make fast informed decisions

Data Processing Pipeline Example

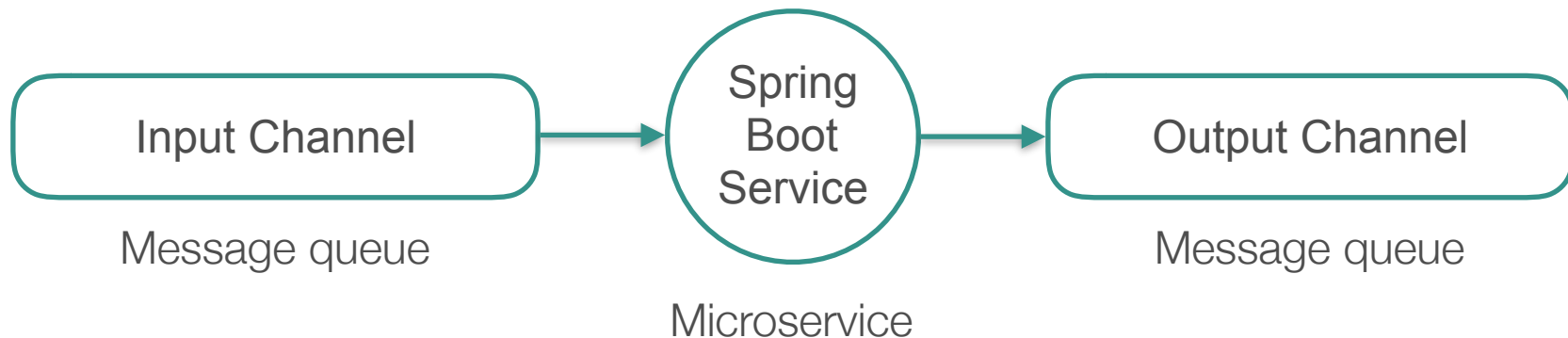


Scalable distributed data processing

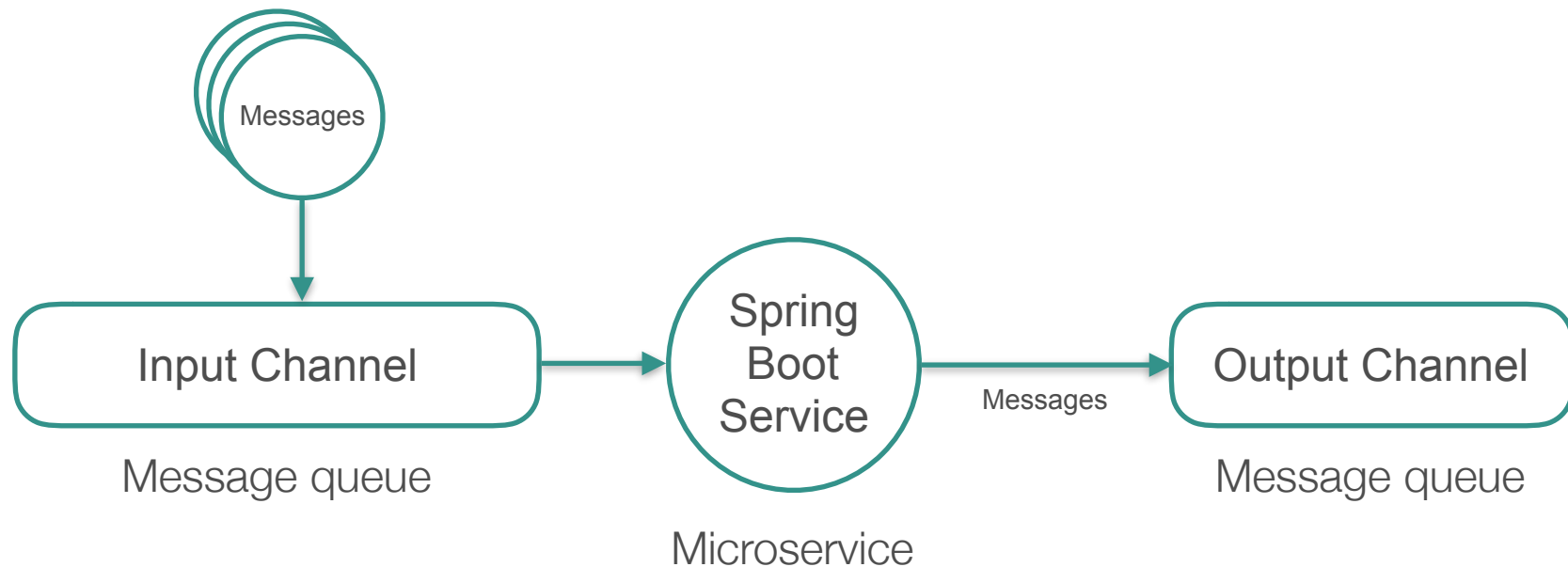


Input and output channels

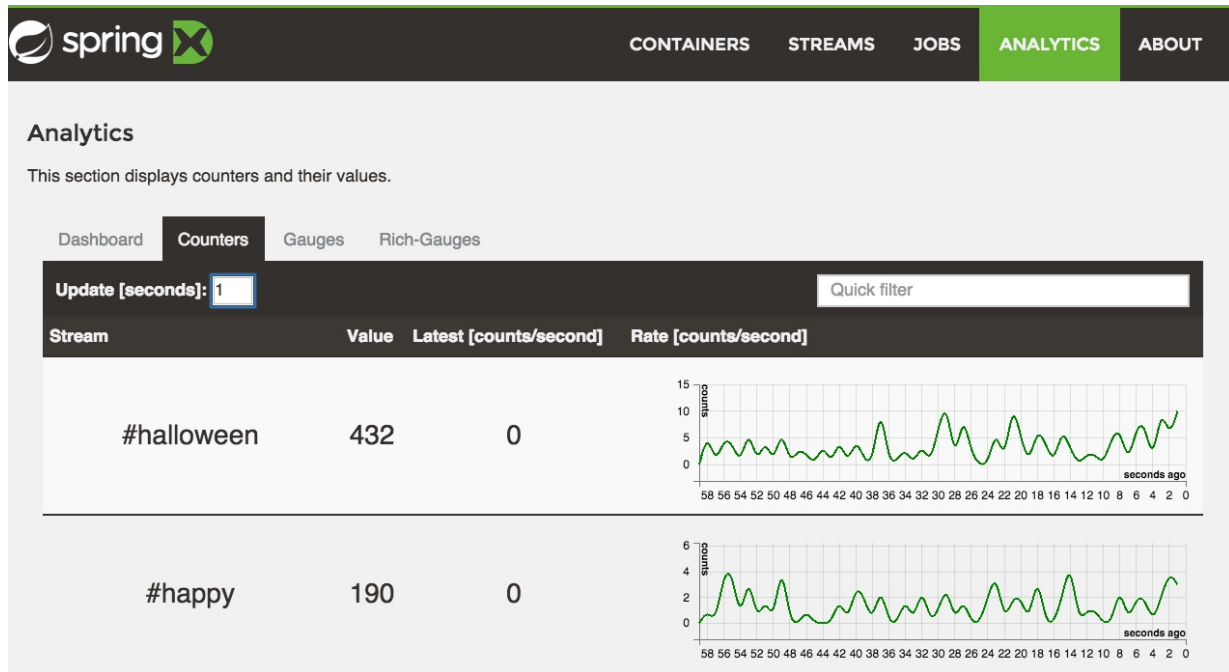
- Each Spring Boot microservice has an input channel and an output channel



Input and output channels

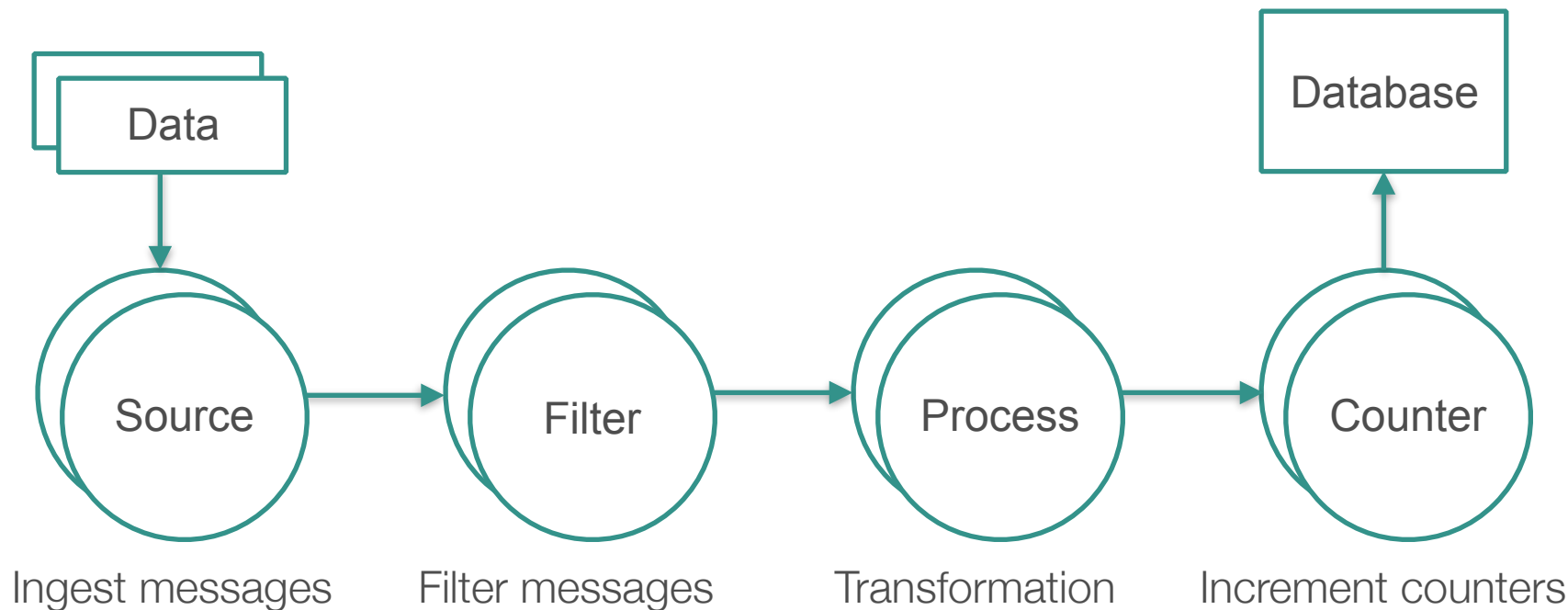


Building Real-time Analytics on Twitter hash tags



Setting up a data processing pipeline

We want to build something like this:



Tweet Source Module

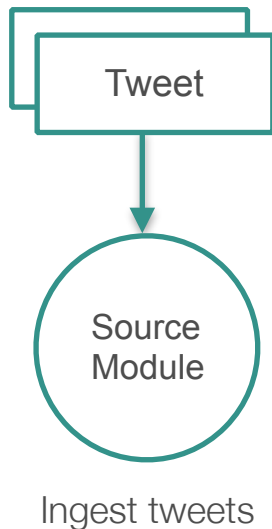
Tweet Source Module

Responsibility of a source module

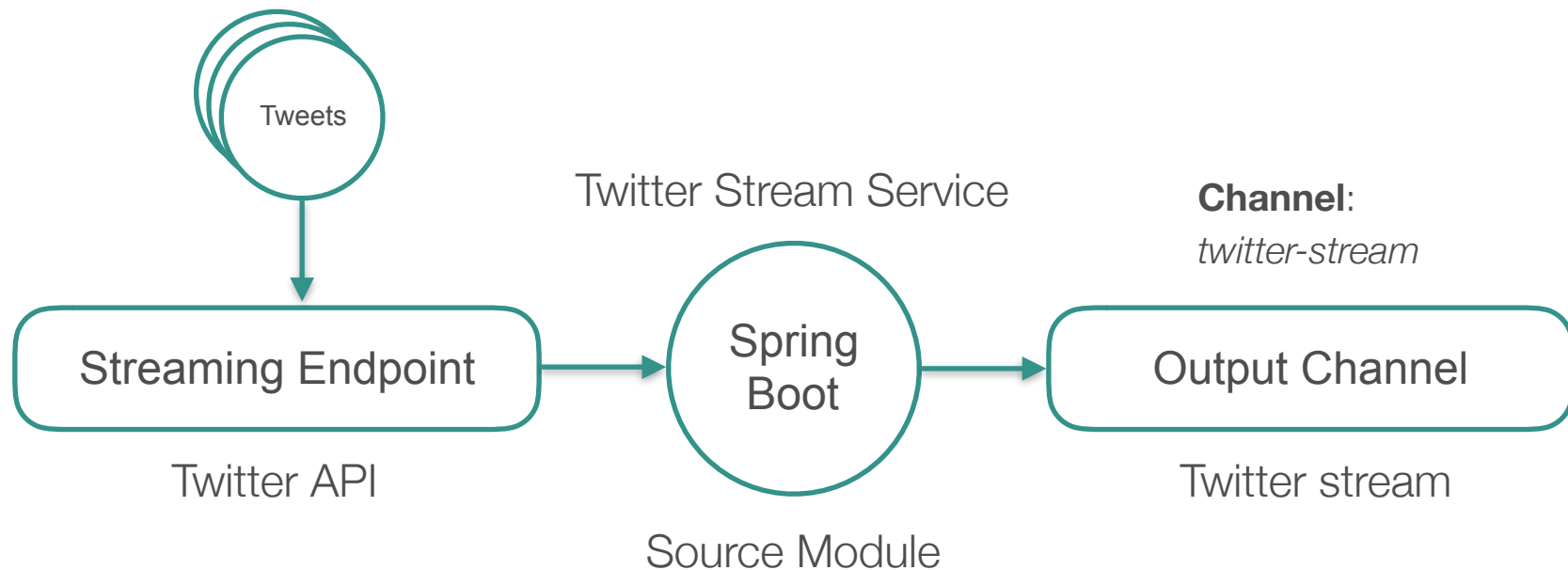
- Ingest data from multiple sources, such as a streaming REST API endpoint or HDFS
- Transform a stream into discrete messages that are uniformly distributed, such as an individual tweet
- Output those messages to an output channel for the next service to process

Ingesting tweets

- We start by building a Spring Boot source module that imports tweets from Twitter



Visualize the tweet source module

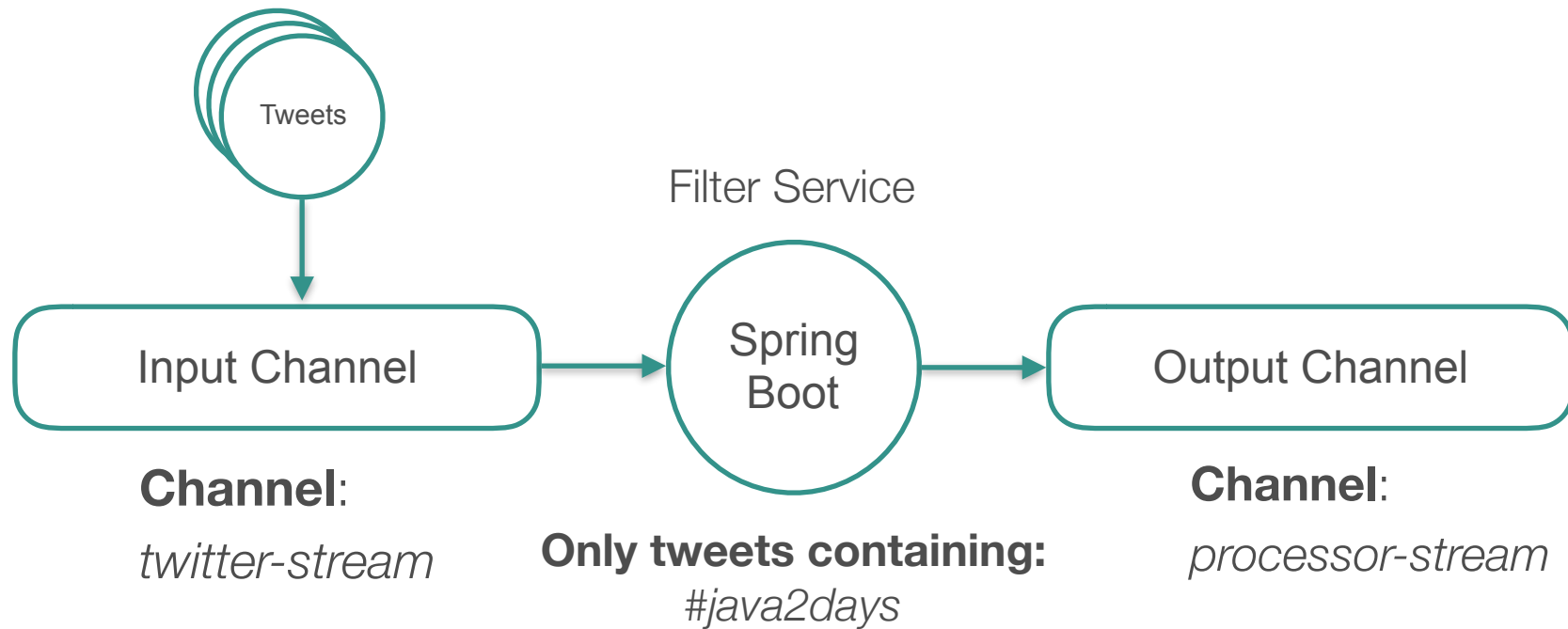


Adding a filter module

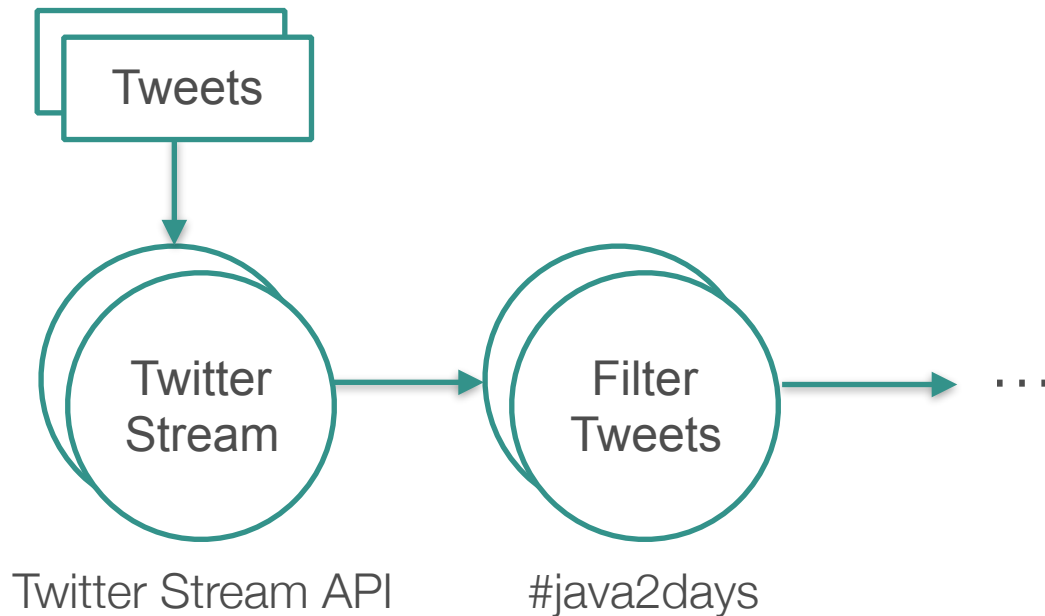
Responsibility of a filter module

- Filter messages from the source module
- Filters noise to increase quality of measurements in downstream modules
- Example:
 - *I only want to measure tweets containing the hash tag **#java2days***

Visualizing the filter module



Our pipeline now looks like:

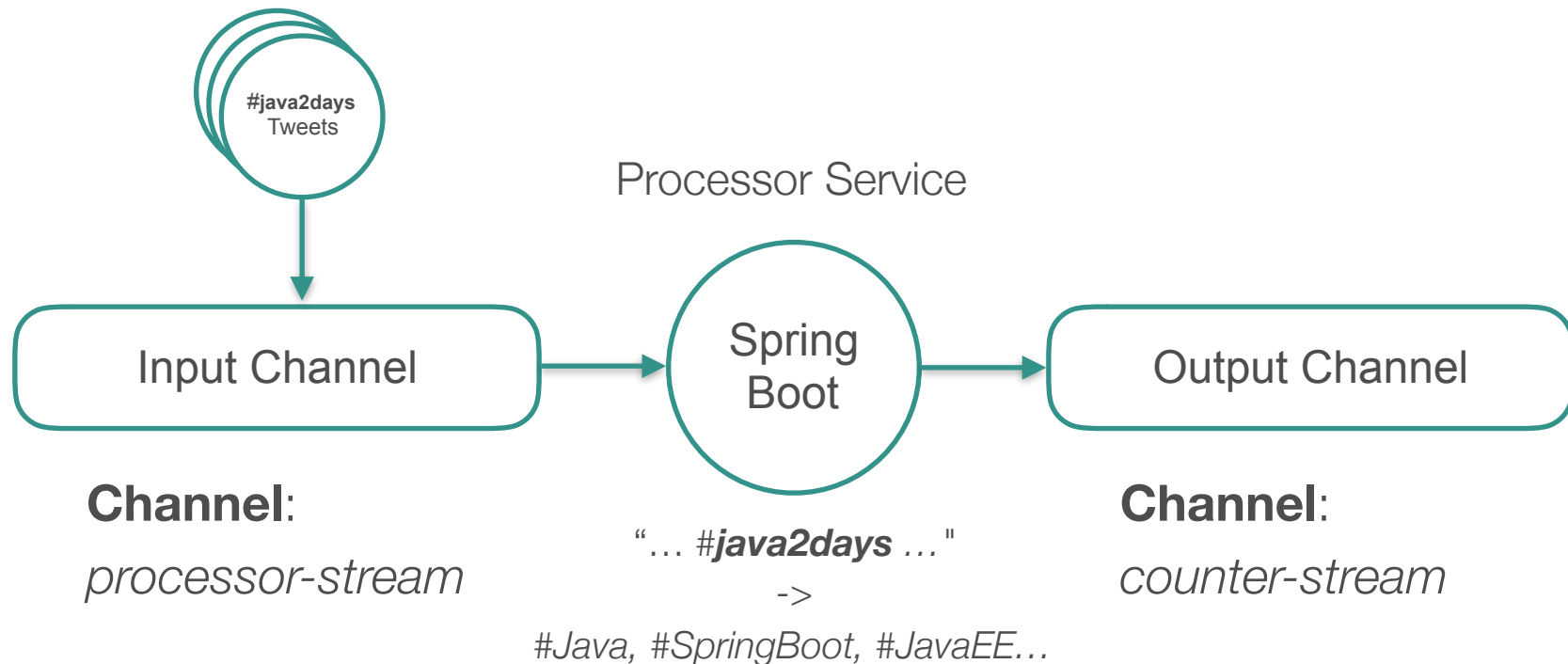


Adding a processor module

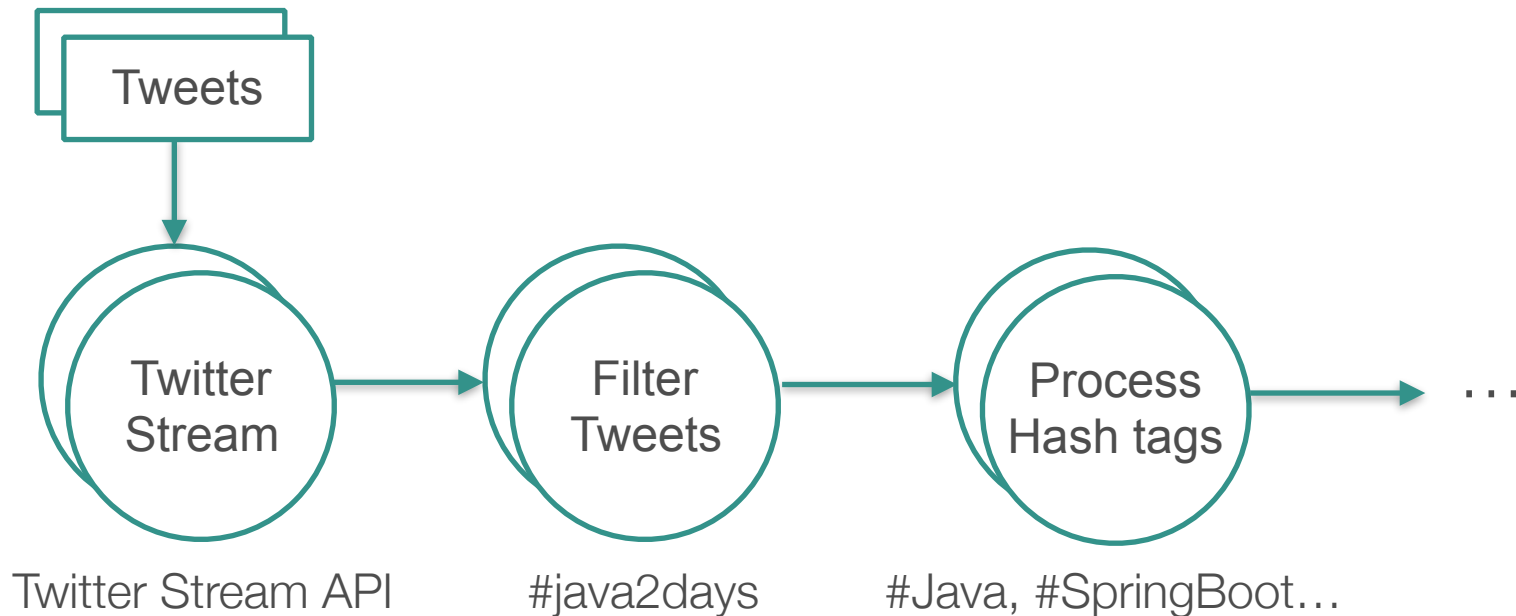
Responsibility of a processor module

- Take a filtered stream of messages and produce multiple output messages by transforming the payload into multiple dimensions of attributes
- For example:
 - Take a **#java2days** tweet and parse the other hash tags and output one message per hash tag
 - **#java2days** -> (*#Java*, *#SpringBoot*, *#JavaEE*)

Processor module



Our pipeline now looks like:

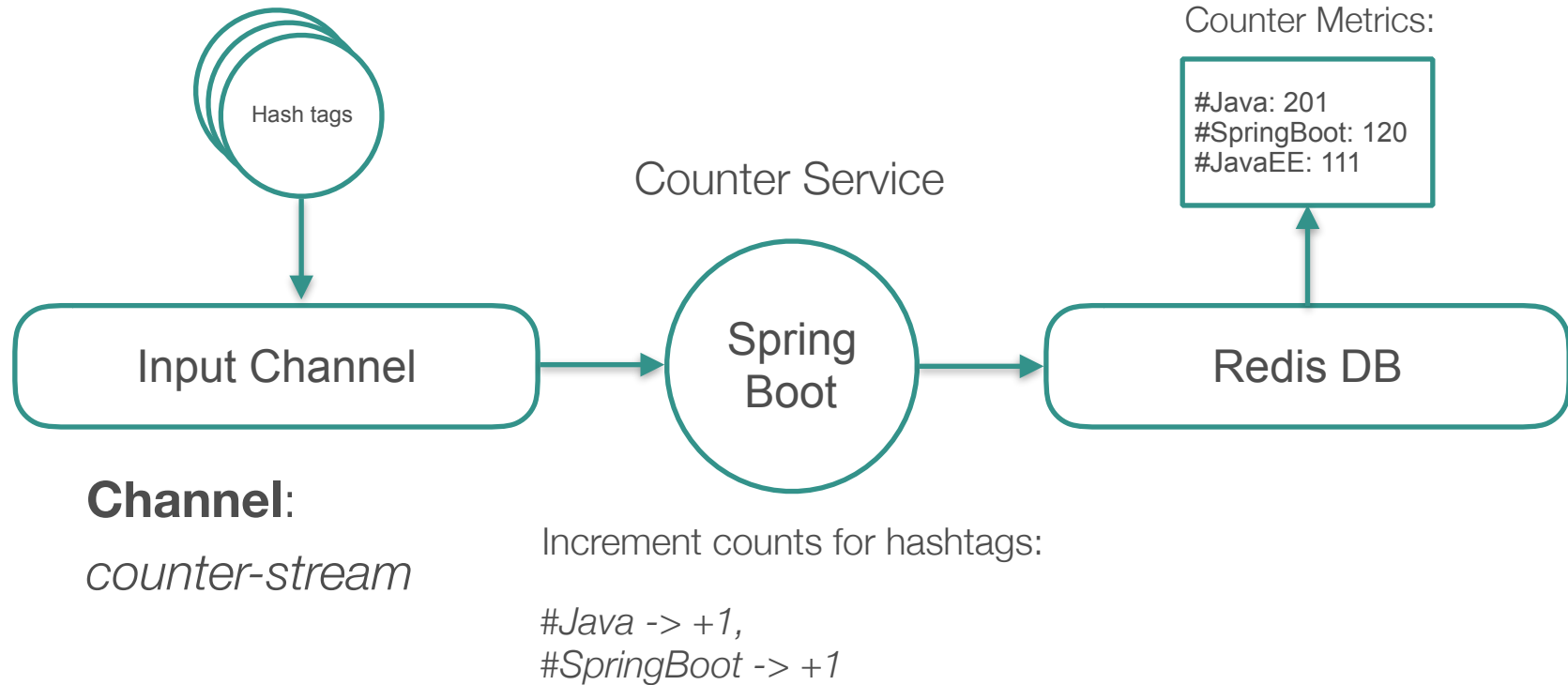


Adding a counter module

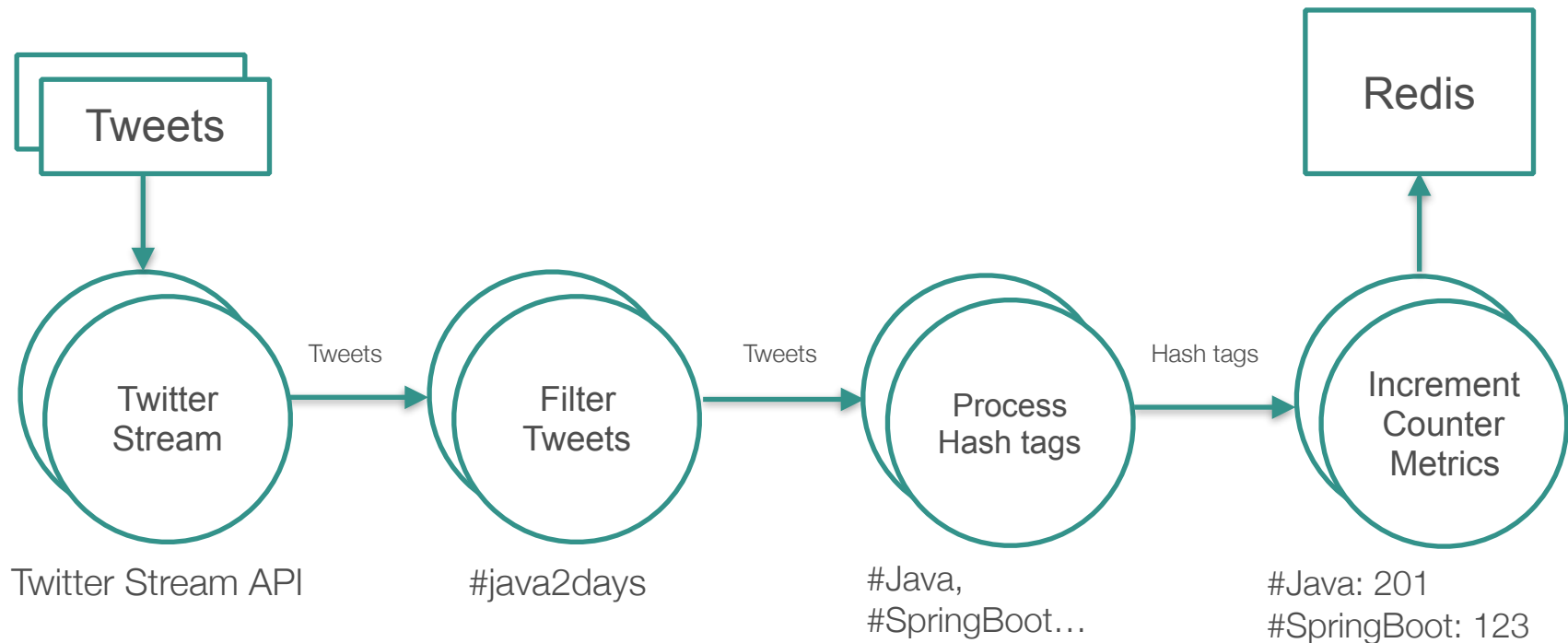
Responsibility of a counter module

- Take messages from an input channel and output an increment to multiple buckets that count message attributes over time
- Save the results to a sink, for example a Redis database
- Use Spring Cloud Data Flow admin tool to measure tweets over time

Counter module



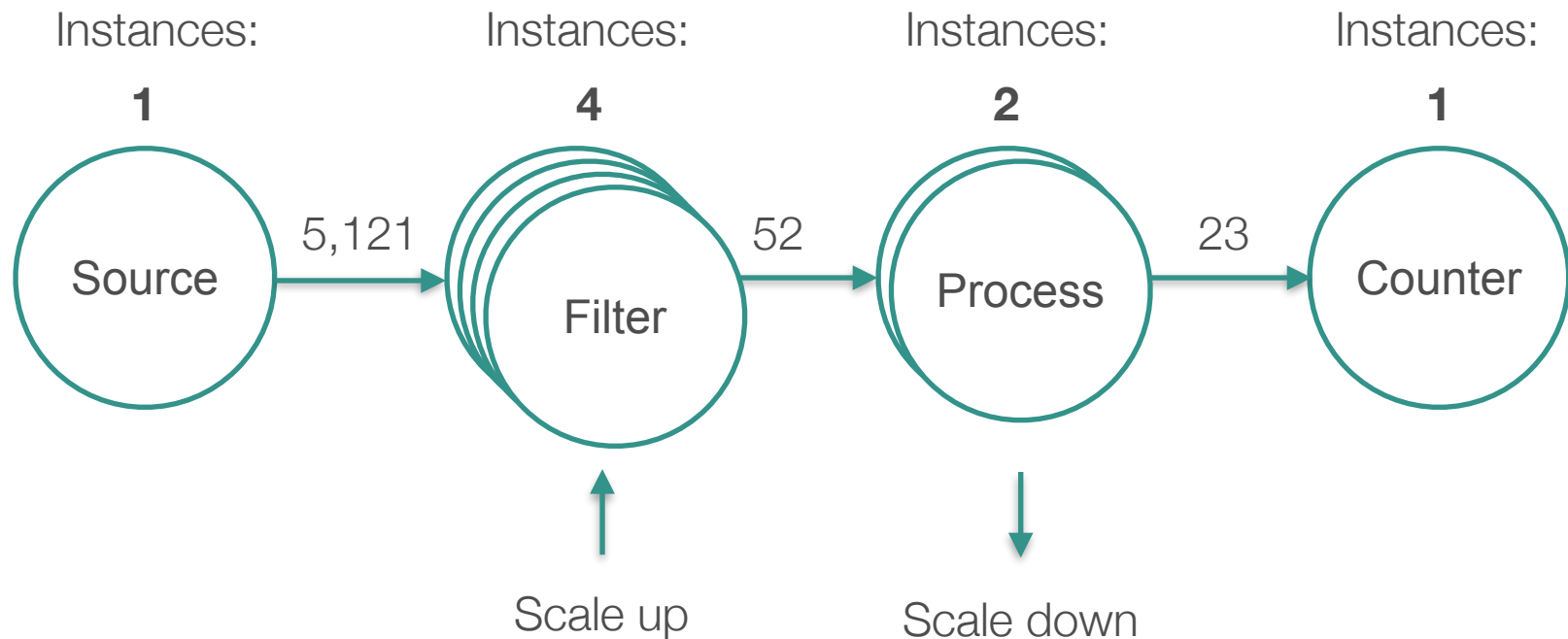
Our pipeline now looks like:



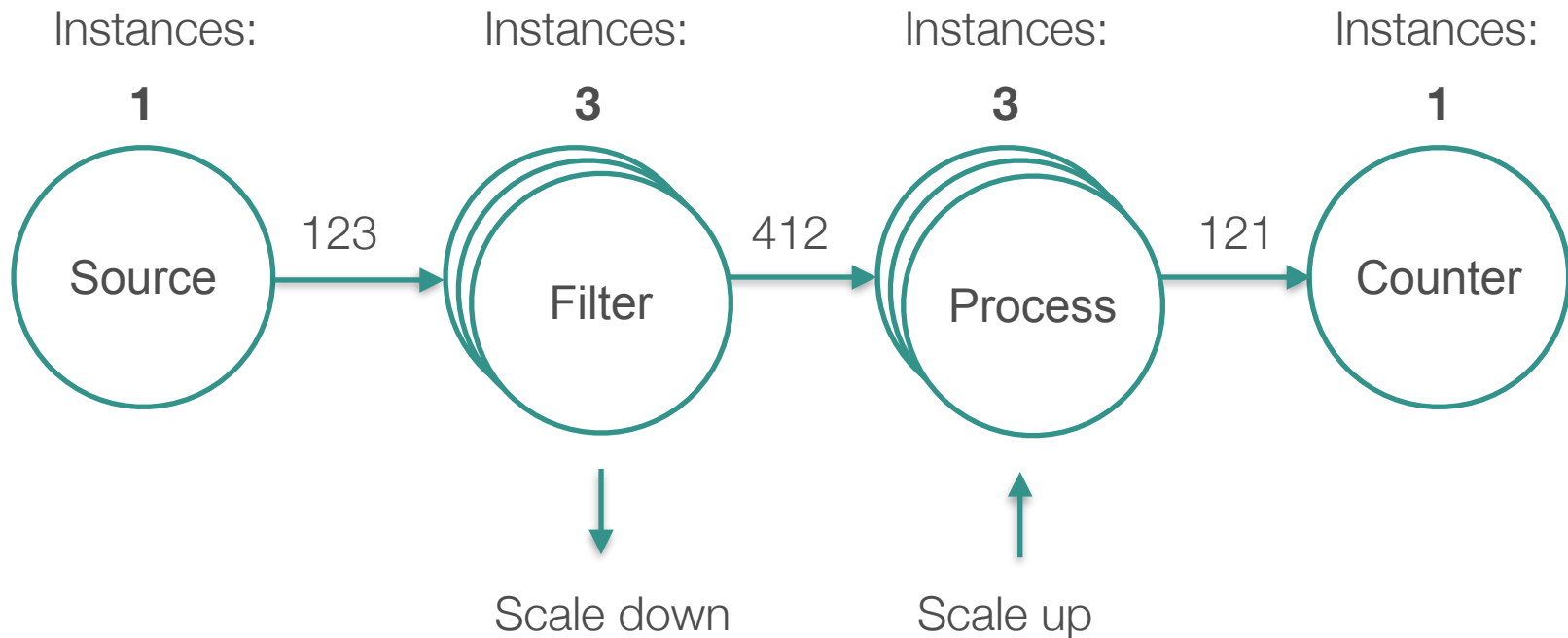
Scaling our pipeline

- Services in the pipeline can be scaled up and down automatically to handle the load and prevent bottlenecks

Auto-scaling



Auto-scaling



Goal of auto-scaling

- Keep the data processing pipeline uniform to prevent bottlenecks
- Optimize the instance count on cloud providers so that cost can be predicted and optimized

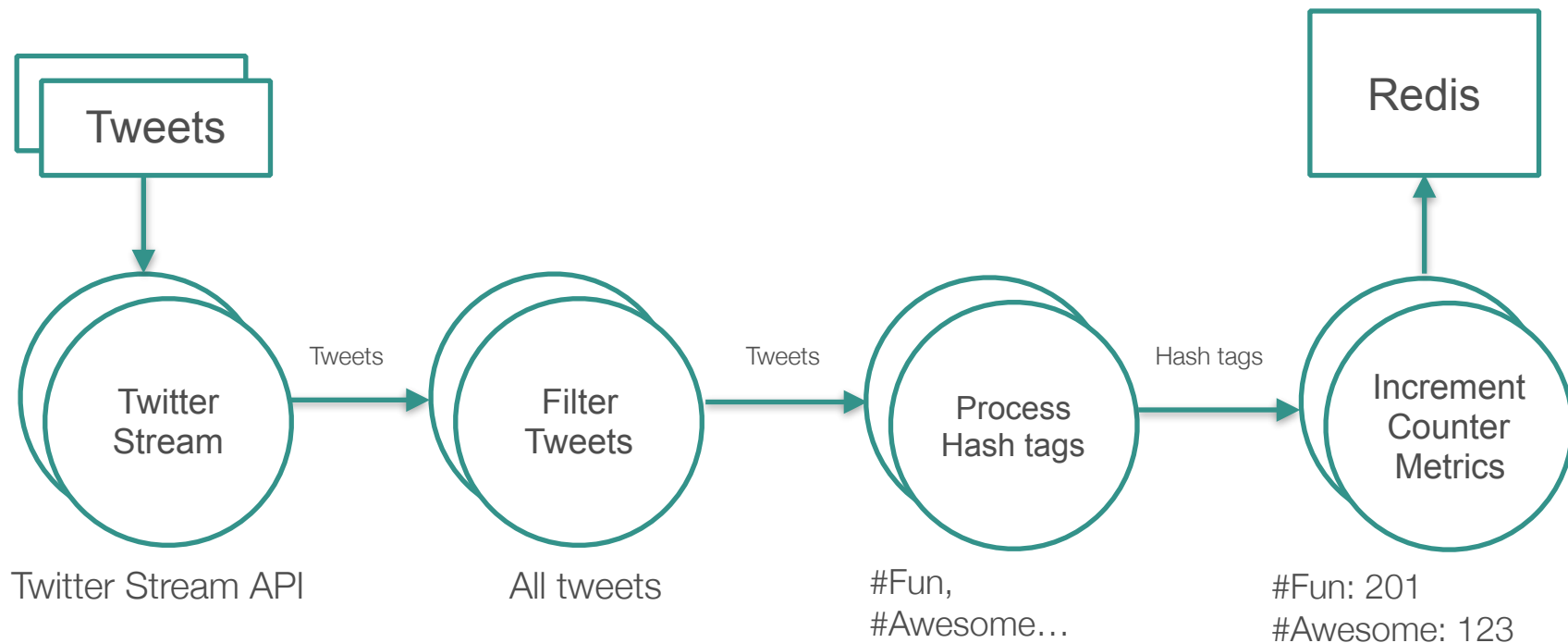
Sinking messages into counters

- Each message in the pipeline has an opportunity to increase multiple counters
- Counters are like buckets, and we can increment those buckets with a **name** and **timestamp**

Twitter Analytics Demo

- We're going to brave the demo gods
- Wish me luck
- First, let's review the steps for the demo (in case it fails)

What we will demo:



Let's go through the steps

- Start a **Redis server**
- Start the **twitter streaming module**
- Start the **filter module**
- Start the **processor module**
- Start the **counter module**
- Start **Spring Cloud Data Flow Admin UI**

Start Redis Server

```
➔ ~ redis-server
50143:C 02 Nov 12:52:50.237 # Warning: no config file specified, using the default config.
path/to/redis.conf
50143:M 02 Nov 12:52:50.239 * Increased maximum number of open files to 10032 (it was origi

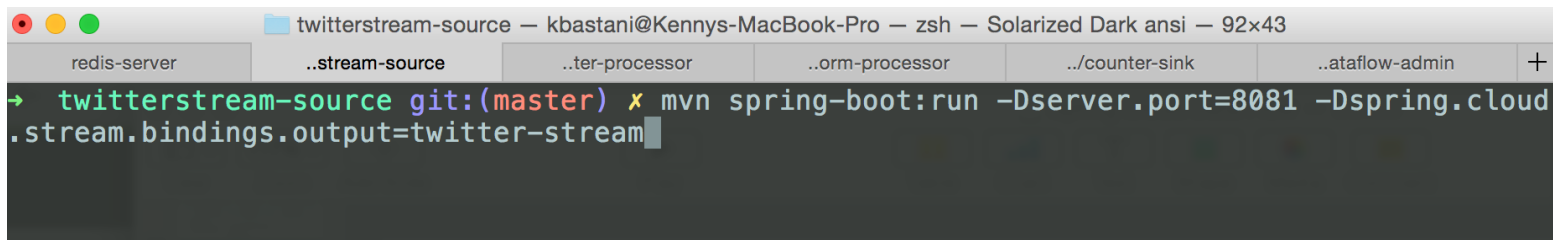
Redis 3.0.4 (00000000/0) 64 bit

Running in standalone mode
Port: 6379
PID: 50143

http://redis.io

50143:M 02 Nov 12:52:50.239 # Server started, Redis version 3.0.4
50143:M 02 Nov 12:52:50.239 * The server is now ready to accept connections on port 6379
```

Start Twitter Stream Module



A terminal window titled "twitterstream-source — kbastani@Kennys-MacBook-Pro — zsh — Solarized Dark ansi — 92x43". The window has several tabs: "redis-server", "..stream-source", "..ter-processor", "..orm-processor", "../counter-sink", and "..ataflow-admin". The active tab is "..stream-source". The terminal shows the command: `→ twitterstream-source git:(master) x mvn spring-boot:run -Dserver.port=8081 -Dspring.cloud.stream.bindings.output=twitter-stream` with a cursor at the end of the line.

Start the Filter Module

redis-server	..stream-source	..ter-processor	..orm-processor	../counter-sink	..ataflow-admin	+
--------------	-----------------	-----------------	-----------------	-----------------	-----------------	---

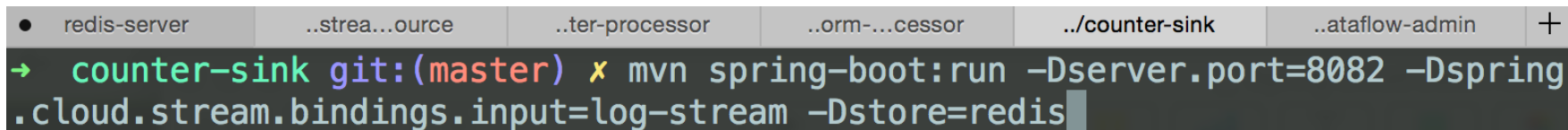
```
→ filter-processor git:(master) ✕ mvn spring-boot:run -Dserver.port=8085 -Dspring.cloud.stream.bindings.input=twitter-stream -Dexpression="payload.toLowerCase().length() > 10" -Dspring.cloud.stream.bindings.output=processor-stream
```

Start the Processor Module

● redis-server	..strea...ource	..ter-processor	..orm-...essor	../counter-sink	..ataflow-admin	+
----------------	-----------------	-----------------	----------------	-----------------	-----------------	---

```
→ transform-processor git:(master) ✕ mvn spring-boot:run -Dserver.port=8084 -Dspring.cloud.stream.bindings.input=processor-stream -Dexpression="new com.fasterxml.jackson.databind.ObjectMapper().readValue(payload,T(java.util.HashMap)).get('text')" -Dspring.cloud.stream.bindings.output=log-stream
```

Start the Counter Module

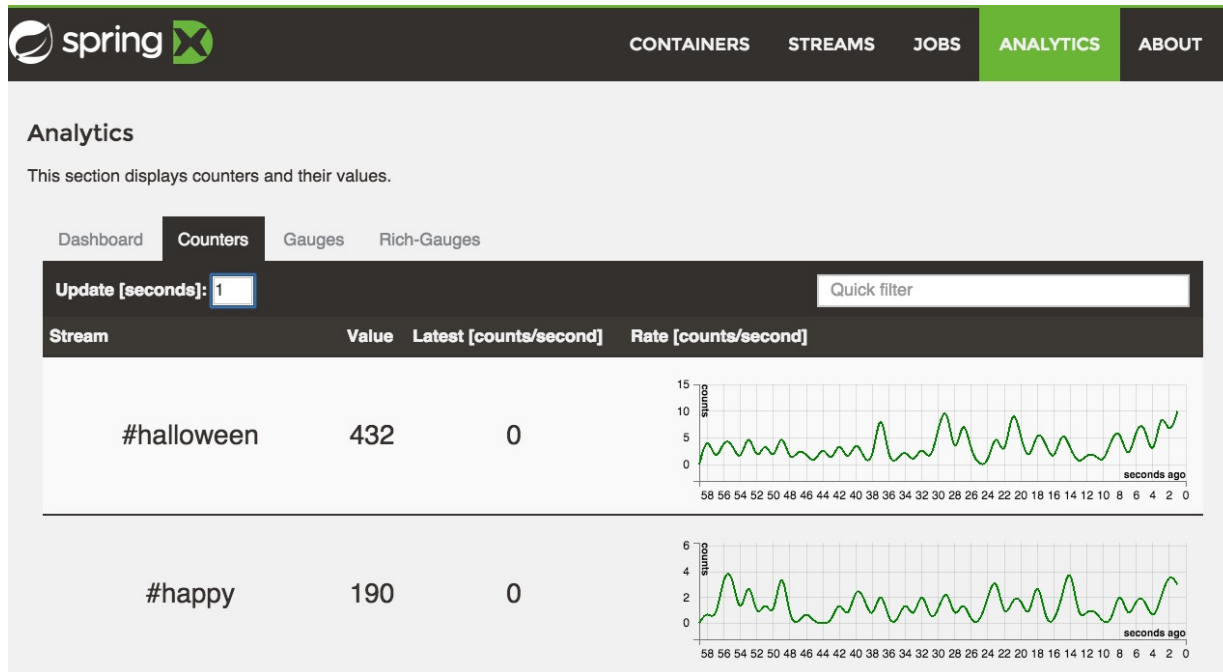


A screenshot of a terminal window with a tabbed interface. The tabs are labeled: 'redis-server', '..strea...ource', '..ter-processor', '..orm-...cessor', './counter-sink', and '..ataflow-admin'. The active tab is './counter-sink'. The terminal shows a command being executed: `→ counter-sink git:(master) x mvn spring-boot:run -Dserver.port=8082 -Dspring.cloud.stream.bindings.input=log-stream -Dstore=redis`. The cursor is at the end of the command line.

Start Spring Cloud Data Flow Admin UI

redis-server	..stream-source	..ter-processor	..orm-processor	../counter-sink	..ataflow-admin
<pre>→ spring-cloud-dataflow-admin git:(master) ✕ mvn spring-boot:run</pre>					

Navigate to the Admin UI



Thanks!

Questions?

<http://start.spring.io/>